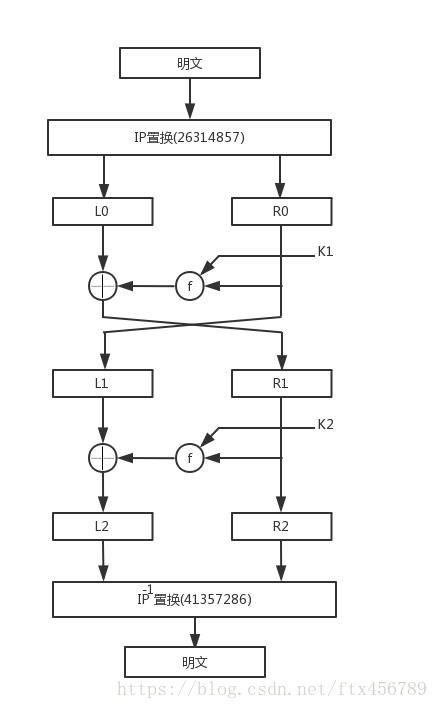
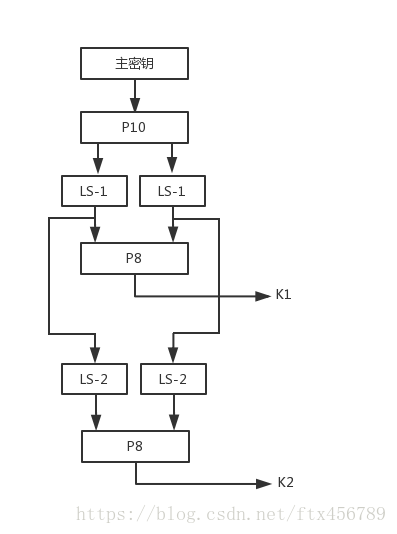
# 开发手册

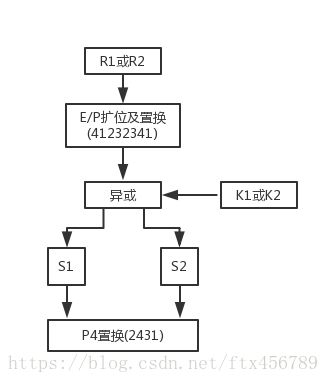
1. **算法简介**

S-DES（Simplified Data Encryption Standard）是一种简化的数据加密标准算法，用于数据的加密和解密。该算法适用于小规模数据加密，并可以使用8位密钥进行加密和解密操作。

基本流程如下：







加密算法：[C=IP^{-1}(f\_{k\_{2}}(SW(f\_{k\_{1}}(IP(P)))))](" \l ")

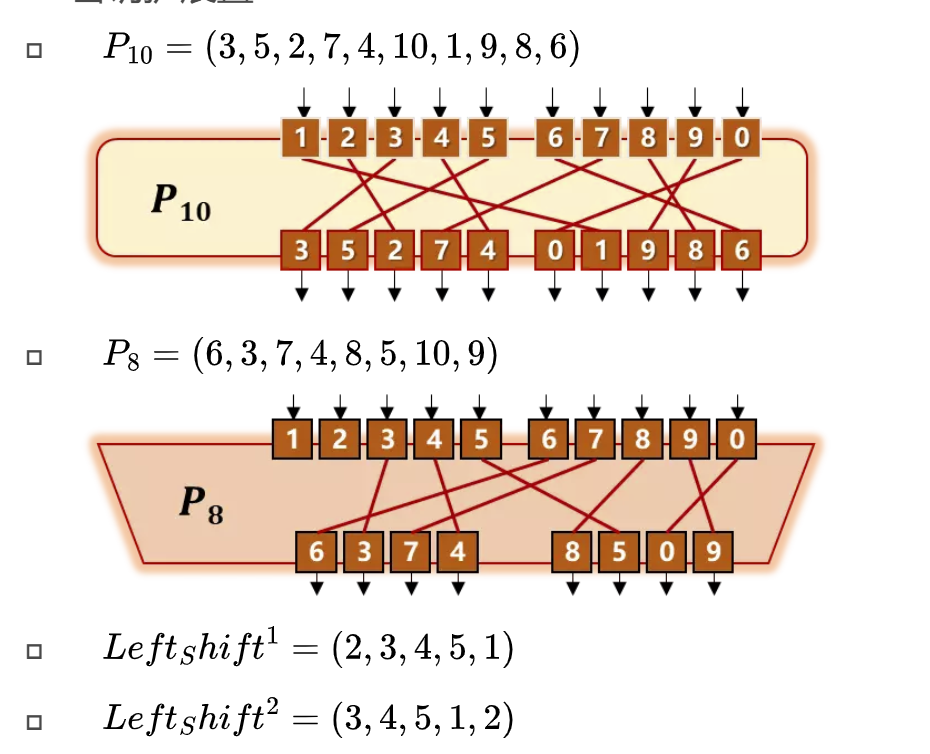
解密算法：[P=IP^{-1}(f\_{k\_{1}}(SW(f\_{k\_{2}}(IP(C)))))](" \l ")

密钥扩展：[k\_{i}=P\_{8}(Shift^{i}(P\_{10}(K))), (i=1,2)](" \l ")

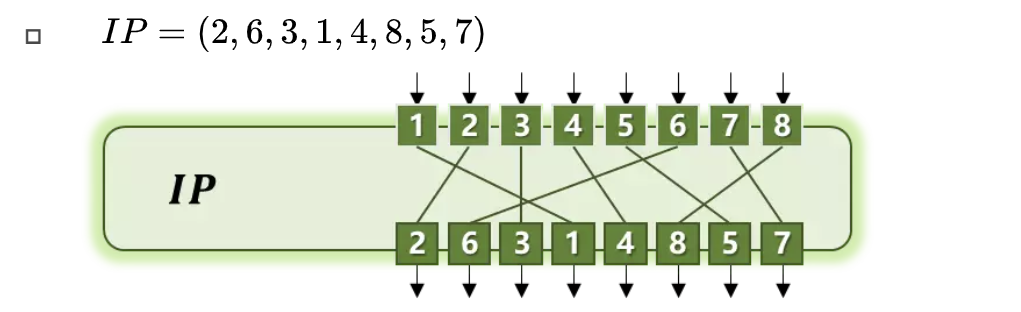
1. **功能需求分析**

**2.1基本装置设定**

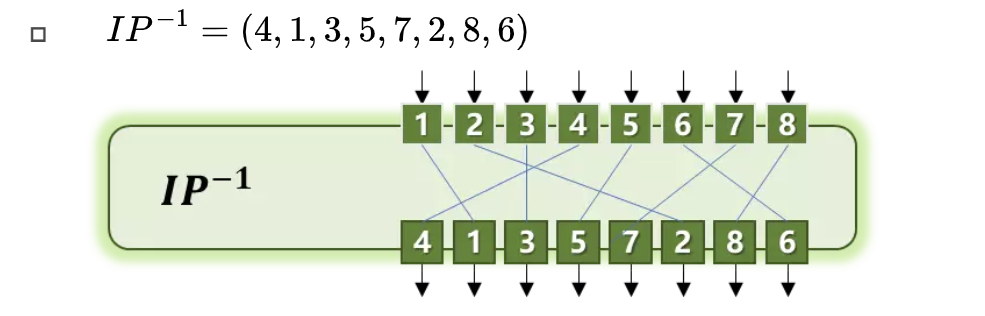
2.1.1密钥扩展置



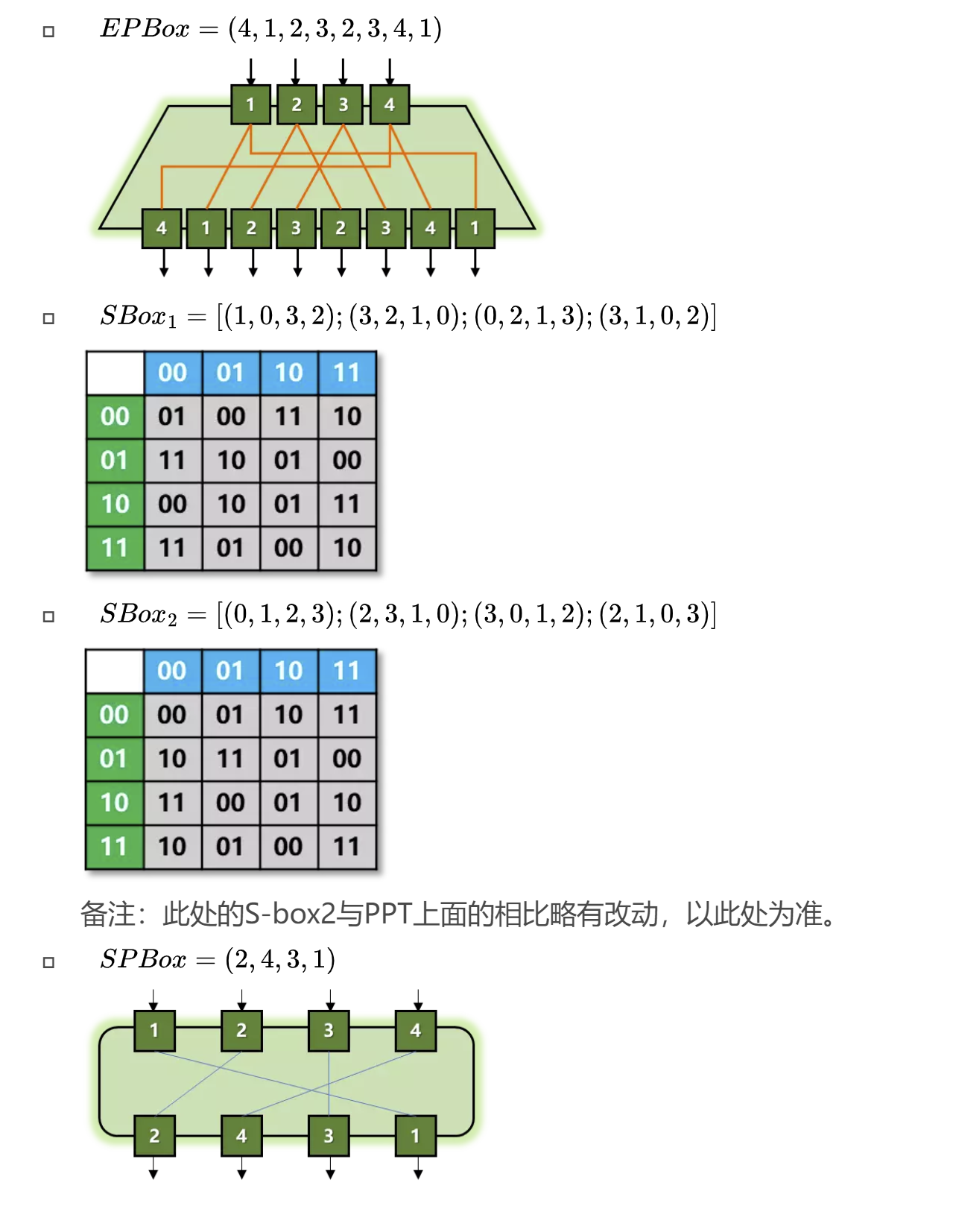
2.1.2初始置换盒



2.1.3最终置换盒



2.1.4轮函数F



**2.2基本功能实现**

根据S-DES算法编写和调试程序，提供GUI解密支持用户交互。输入可以是8bit的数据和10bit的密钥，输出是8bit的密文。

且考虑到**算法标准**，编写程序的时候需要使用相同算法流程和转换单元(P-Box、S-Box等)，以保证算法和程序在异构的系统或平台上都可以正常运行。

**2.3扩展功能和破解算法**

2.3.1扩展功能

加密算法的数据输入可以是ASII编码字符串(分组为1 Byte)，对应地输出也可以是ACII字符串(很可能是乱码)

2.3.2破解算法

使用相同密钥的明、密文对(一个或多个)，可通过暴力破解的方法找到正确的密钥Key。在编写程序时，使用多线程的方式提升破解的效率。输出所有可用的密钥，并且记录破解时间并展示。

1. **设计实现**

3.1基本功能实现与扩展功能

Js部分代码：

var DES = {

    text: [],//明文

    stext: [],//密文

    key: [],

    key0: [],//记录初始密钥

    P10Box: [3, 5, 2, 7, 4, 10, 1, 9, 8, 6],

    P8Box: [6, 3, 7, 4, 8, 5, 10, 9],

    LeftShift1Box: [2, 3, 4, 5, 1],

    LeftShift2Box: [3, 4, 5, 1, 2],

    IP1Box: [2, 6, 3, 1, 4, 8, 5, 7],

    IP2Box: [4, 1, 3, 5, 7, 2, 8, 6],

    EPbox: [4, 1, 2, 3, 2, 3, 4, 1],

    SPBox: [2, 4, 3, 1],

    SBox1: [[1, 0, 3, 2], [3, 2, 1, 0], [0, 2, 1, 3], [3, 1, 0, 2]],

    SBox2: [[0, 1, 2, 3], [2, 3, 1, 0], [3, 0, 1, 2], [2, 1, 0, 3]],

    //密钥置换

    P10: function (arr) {

        let temp = [];

        temp = arr.concat();

        for (let i = 0; i < arr.length; i++) {

            temp[i] = arr[this.P10Box[i] - 1];

        }

        this.key = temp.concat();

    },

    P8: function (arr) {

        let temp = [];

        temp = arr.slice(2);

        for (let i = 0; i < arr.length; i++) {

            temp[i] = arr[this.P8Box[i] - 1];

        }

        this.key = temp.slice(0, 8);

    },

    LeftShift1: function (arr) {

        let temp = [];

        temp = arr.concat();

        for (let i = 0; i < arr.length; i++) {

            temp[i] = arr[this.LeftShift1Box[i] - 1];

        }

        return temp;

    },

    LeftShift2: function (arr) {

        let temp = [];

        temp = arr.concat();

        for (let i = 0; i < arr.length; i++) {

            temp[i] = arr[this.LeftShift2Box[i] - 1];

        }

        return temp;

    },

    //f函数

    Ffuc: function (R, k) {

        let temp = [], left = [], right = [];

        for (let i = 0; i < 8; i++) {

            temp[i] = R[this.EPbox[i] - 1] ^ k[i];

        }

        left = temp.slice(0, 4);

        right = temp.slice(4);

        let leftrow = left[0] \* 2 + left[3];

        let leftcol = left[1] \* 2 + left[2];

        let rightrow = right[0] \* 2 + right[3];

        let rightcol = right[1] \* 2 + right[2];

        left[3] = (this.SBox1[leftrow][leftcol] / 2);//0

        left[0] = this.SBox1[leftrow][leftcol] % 2;//1

        left[2] = this.SBox2[rightrow][rightcol] / 2;//2

        left[1] = this.SBox2[rightrow][rightcol] % 2;//3

        return left.slice(0, 4);

    },

    //初始置换

    IP1: function (arr) {

        let temp = [];

        temp = arr.concat();

        for (let i = 0; i < arr.length; i++) {

            arr[i] = temp[this.IP1Box[i] - 1];

        }

    },

    IP2: function (arr) {

        let temp = [];

        temp = arr.concat();

        for (let i = 0; i < arr.length; i++) {

            arr[i] = temp[this.IP2Box[i] - 1];

        }

    }

}

function XOR(a, b) {

    let c = [];

    for (let i = 0; i < a.length; i++)

        c[i] = a[i] ^ b[i];

    return c;

}

function keymake1(arr) {

    DES.P10(arr);

    let key1 = DES.LeftShift1(DES.key.slice(0, 5));

    let key2 = DES.LeftShift1(DES.key.slice(5));

    DES.P8(key1.concat(key2));

}

function keymake2(arr) {

    DES.P10(arr);

    let key1 = DES.LeftShift2(DES.key.slice(0, 5));

    let key2 = DES.LeftShift2(DES.key.slice(5));

    DES.P8(key1.concat(key2));

}

function SDES() {

    DES.IP1(DES.text);

    let L = DES.text.slice(0, 4);

    let R = DES.text.slice(4);

    let mykey = DES.key;

    keymake1(mykey);

    let f = DES.Ffuc(R, DES.key)

    L = XOR(L, f);

    let temp = []; temp = L; L = R; R = temp;

    keymake2(mykey);

    f = DES.Ffuc(R, DES.key)

    L = XOR(L, f);

    DES.stext = L.concat(R);

    DES.IP2(DES.stext);

}

//解密函数

function SDES2() {

    DES.IP1(DES.stext);

    let L = DES.stext.slice(0, 4);

    let R = DES.stext.slice(4);

    let mykey = DES.key;

    keymake2(mykey);

    let f = DES.Ffuc(R, DES.key)

    L = XOR(L, f);

    let temp = []; temp = L; L = R; R = temp;

    keymake1(mykey);

    f = DES.Ffuc(R, DES.key)

    L = XOR(L, f);

    DES.text = L.concat(R);

    DES.IP2(DES.text);

}

//先定义一个空数组装准备输入的数据

//连接、定义输入框，提交按钮和显示框

var dPut = document.getElementById("put");

var dBtn = document.getElementById("btn");

var dInt = document.getElementById("int");

//点击按钮实现功能

dBtn.onclick = function array() {

    let tt = dPut.value;

    if (tt.length == 8 && DES.key.length == 10) {

        for (let n of tt) {

            DES.text.push(n);

        }

        SDES();

        let s = "密文为： ";

        for (let i of DES.text)

            s += i;

        dInt.value = s;

        DES.text = [];

        DES.stext = [];

        DES.key = DES.key0.concat();//恢复密钥

    }

    else

        alert("请输入10bit密钥和8bit明文")

}

//输入密钥

var dPut2 = document.getElementById("put2");

var dBtn2 = document.getElementById("btn2");

dBtn2.onclick = function array() {

    DES.key = [];

    let tt = dPut2.value;

    if (tt.length == 10) {

        for (let n of tt) {

            DES.key.push(n);

        }

        DES.key0 = DES.key.concat();

    }

    else

        alert("请输入10bit密钥")

}

//解密

var dPut3 = document.getElementById("put3");

var dBtn3 = document.getElementById("btn3");

dBtn3.onclick = function array() {

    let tt = dPut3.value;

    if (tt.length == 8 && DES.key.length == 10) {

        for (let n of tt) {

            DES.stext.push(n);

        }

        //解密函数,解密后明文为DES.text

        SDES2();

        dInt.value = "";

        //输出明文

        let s = "明文为： ";

        for (let i of DES.text)

            s += i;

        dInt.value = s;

        DES.text = [];

        DES.stext = [];

        DES.key = DES.key0.concat();//恢复密钥

    }

    else

        alert("请输入10bit密钥和8bit密文")

}

3.2破解算法实现

C++代码：

#include<bits/stdc++.h>

#include <stdio.h>

#include <string.h>

#include<iostream>

using namespace std;

#define FFI(a, b) for(int i = a; i < b; i++)

#define FFJ(a, b) for(int j = a; j < b; j++)

#define FFK(a, b) for(int k = a; k < b; k++)

#define RR(a, b) for(int i = a; i > b; i++)

#define ME(a, b) memset(a, b, sizeof(a))

#define SC(x) scanf("%d", &x)

#define SCC(a, b) scanf("%d%d", &a, &b)

#define SCCC(a, b, c) scanf("%d%d%d", &a, &b, &c)

#define PR(x) printf("%d\n", x)

#define PRR(a, b) printf("%d%d\n", a, b);

#define INF 0x3f3f3f3f

#define MAX 50

#define MOD 1000000007

#define E 2.71828182845

#define M 8

#define N 6

typedef long long LL;

const double PI = acos(-1.0);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int k1 = 0;int k2 = 0;

static int P10MAX = 10;

static int P8MAX = 10;

static int P4MAX = 4;

static int IPMAX = 8;

static int IPIMAX = 8;

static int EPMAX = 4;

static int P10[] = {3, 5, 2, 7, 4, 10, 1, 9, 8, 6, '\0'};

static int P8[] = {6, 3, 7, 4, 8, 5, 10, 9, '\0'};

static int P4[] = {2, 4, 3, 1, '\0'};

static int IP[] = {2, 6, 3, 1, 4, 8, 5, 7, '\0'};

static int IPI[] = {4, 1, 3, 5, 7, 2, 8, 6, '\0'};

static int EP[] = {4, 1, 2, 3, 2, 3, 4, 1, '\0'};

static int S0[4][4] = {

{1, 0, 3, 2},

{3, 2, 1, 0},

{0, 2, 1, 3},

{3, 1, 3, 2},

};

static int S1[4][4] = {

{0, 1, 2, 3},

{2, 0, 1, 3},

{3, 0, 1, 0},

{2, 1, 0, 3},

};

//将二进制变成数字

int BinaryToDecimal(string binary){

int result = 0;

FFI(0, binary.length())result = 2 \* result + (binary[i] - '0');

return result;

}

//数组的长度

int Getlength(int p[]){

int i = 0;

for(; p[i] != '\0'; i++);

return i;

}

//置换函数

int Permute(int inum, int p[], int pmax){

int result = 0, length = Getlength(p);

FFI(0, length){

result <<= 1;

result |= (inum >> (pmax - p[i])) & 1;

}

return result;

}

void SDES(int k){

int t1 ,t2;

t1 = t2 = 0;

k = Permute(k, P10, P10MAX);

//保留最低5位

t1 = (k >> 5) & 0x1f;

t2 = k & 0x1f;

t1 = ((t1 & 0xf) << 1) | ((t1 & 0x10) >> 4);

t2 = ((t2 & 0xf) << 1) | ((t2 & 0x10) >> 4);

k1 = Permute((t1 << 5) | t2, P8, P8MAX);

t1 = ((t1 & 0x07) << 2) | ((t1 & 0x18) >> 3);

t2 = ((t2 & 0x07) << 2) | ((t2 & 0x18) >> 3);

k2 = Permute((t1 << 5) | t2, P8, P8MAX);

}

//fk里面的F函数

int F(int R, int K){

int t = Permute(R, EP, EPMAX) ^ K;

int t0 = (t >> 4) & 0xf;

int t1 = t & 0xf;

t0 = S0[((t0 & 0x8) >> 2) | (t0 & 1)][(t0 >> 1 & 0x3)];

t1 = S1[((t1 & 0x8) >> 2) | (t1 & 1)][(t1 >> 1 & 0x3)];

t = Permute((t0 << 2) | t1, P4, P4MAX);

return t;

}

//最低4位

int fk(int m, int k){

int l = (m >> 4) & 0xf;

int r = m & 0xf;

return ((l ^ F(r, k)) << 4) | r;

}

int SW(int x){

return ((x & 0xf) << 4) | ((x >> 4) & 0xf);

}

//加密

int encrypt(int m){

m = Permute(m, IP, IPMAX);

m = fk(m, k1);

m = SW(m);

m = fk(m, k2);

m = Permute(m, IPI, IPIMAX);

return m;

}

//解密

int decrypt(int m){

m = Permute(m, IP, IPMAX);

m = fk(m, k2);

m = SW(m);

m = fk(m, k1);

m = Permute(m, IPI, IPIMAX);

return m;

}

void printBin(int x, int n){

int mask = 1 << (n - 1);

while(mask > 0){

((x & mask) == 0) ? printf("0") : printf("1");

mask >>= 1;

}

cout<<endl;

}

int main(void){

ios::sync\_with\_stdio(false); cin.tie(0); cout.tie(0);

int index, ciphertext2;string plaintext, ciphertext1, key;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 输入均为二进制 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*请输入明文和密文进行破解\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*请输入明文:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cin>>plaintext;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*请输入密文:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cin>>ciphertext1;

cout<<"可用密钥："<<endl;

auto start = std::chrono::high\_resolution\_clock::now(); // 记录开始时间

for(int i=0;i<=1023;i++)

{

SDES(i);

ciphertext2=encrypt(BinaryToDecimal(ciphertext1));

if(BinaryToDecimal(ciphertext1)==ciphertext2&&i>0)

{

char str[] = "0000000000";

for (int l = 0; l < 10; l++) if (((1 << l) & i) == (1 << l)) str[9 - l] = '1';

printf("%s", str);

cout<<endl;

}

}

cout<<"结束";

auto end = std::chrono::high\_resolution\_clock::now(); // 记录结束时间

std::chrono::duration<double, std::milli> duration = end - start; // 计算执行时间

std::cout << "破解时间：" << duration.count() << " 毫秒" << std::endl;

return EXIT\_SUCCESS;

}

1. **测试结果**

测试结果见测试结果文档。